



# ADRIME COMPONENTS MANUAL

## Table of contents

1	Introduction	4
1.1	About ScreenAds	4
1.1.1	Fixed positions	4
1.1.2	Layers	4
1.1.3	Assets	4
2	Getting started	5
2.1	System requirements	5
2.2	Installation	5
2.3	Using the Adrime Components	5
3	About the components	6
3.1	As Interface component	6
3.2	Button component	6
3.3	Command component	7
3.4	Click component	8
3.5	Clipper component	9
3.6	Positioner component	10
3.6.1	Positioning and aligning a layer	10
3.6.2	Set the size, stage align and the scale mode	12
3.6.3	Other settings: Hide elements And Sticky	13
3.6.4	Moving from one position to another	13
4	Using the ScreenAd Library with Actionscript	14
4.1	The ScreenAd global object	14
4.2	Position a layer with Actionscript	14
4.2.1	Positioning elements in a scaling movie	15
4.2.2	Position transitions	15
4.2.3	Sticky positioning	15
4.3	Clipping a layer in Actionscript	15
4.4	Create an exit click	16
4.4.1	Dynamic clicks	16
4.4.2	Setting a wallpaper	16

4.5	Event tracking	17
4.5.1	Timer events	17
4.6	Getting information on the environment of the ad	17
4.6.1	Getting the mouse position	17
4.6.2	Getting the banner position	18
4.7	Setting up communication between flash elements	18
4.7.1	Shared connection	18
4.7.2	ScreenAd commands	19
4.8	ScreenAd Events and Properties	21
4.8.1	screenad.ENVIRONMENT	22
4.8.2	screenad.RESIZE	22
4.8.3	screenad.PRELOAD_COMPLETE	22
4.8.4	screenad.BUTTON	22
4.8.5	screenad.BUTTON_OUT	23
4.8.6	screenad.BUTTON_OVER	23
4.8.7	screenad.CLICK	23
4.8.8	screenad.CLICK_OVER	23
4.8.9	screenad.CLICK_OUT	23
4.8.10	screenad.VISIBILITY	23
4.8.11	screenad.SHOW	23
4.8.12	screenad.HIDE	23
4.8.13	screenad.MOUSE_LEAVE	24
4.8.14	screenad.shared.NEW_ELEMENT	24
5	Using Flex, FDT or Flashdevelop	25
6	Actionscript reference	26
6.1	Methods	26
6.2	Events	28
6.3	Properties	29
7	Contact	31
7.1	Fixed Support	31

## 1 Introduction

Adrime's free components suite is designed to make it as easy as possible to create Rich Media in Adobe Flash.

The suite integrates easily within Adobe Flash, and enables you to design Rich Media without the trouble of writing your own Actionscript codes. The creatives you provide can be incorporated worldwide within every ad management system, so you don't need to bother about specific website requirements.

Our software is divided into two parts. First, the Adrime Components for building your ScreenAds and the Adrime Online Previewer for previewing and submitting your ScreenAds for usage. This document will focus on the components.

Please feel free to use and distribute this software, also our Adrime Online Previewer. If you have any questions, don't hesitate to contact our support desk.

### 1.1 About ScreenAds

The ScreenAd technology is created by Adrime. ScreenAds are online advertisements that are dynamically and interactively shown over a web page outside the usual banner positions. They are the most effective form of online advertising. Adrime technology makes it easy to create, present and traffic advanced rich media. These advertisements can be placed in layers across the website, in the form of Expandable banners, mouse chasers, fullscreen ads and sticky-ads.

#### 1.1.1 Fixed positions

A fixed position is a Flash or GIF file embedded in the web page. This can be a banner, a skyscraper or any format that fills an advertising position on the page.

#### 1.1.2 Layers

A layer (or layered ad) is a Flash animation that floats over the web page. Layers can have any size or position on the page. The components allow you to align layers to the browser window, the mouse cursor and even the fixed position. Layers have a transparent background by default.

#### 1.1.3 Assets

All files that are loaded by fixed positions or layers are called assets. This can be a Flash video file that is loaded progressively, an image that changes the background of the entire page or a XML data file.

## 2 Getting started

To install the Adrime Components for Adobe Flash, please follow the following steps.

### 2.1 System requirements

The Adrime Components require the following software prior to installation:

- Adobe Flash CS3 (or higher, standard or professional edition)
- Adobe Extensions Manager ( If you don't have the manager installed you can download it free of charge at this location: [http://www.adobe.com/exchange/em\\_download/](http://www.adobe.com/exchange/em_download/))

### 2.2 Installation

1. Download and save the Adrime Components from: <http://clients.adrime.com>
2. Open the Adobe Extension Manager
3. Select Adobe Flash and Remove any old versions of the Adrime Components.
4. Click Install and select the 'Adrime Components.mxp' file.
5. Restart Flash.

### 2.3 Using the Adrime Components

After the installation, you will find the new components in the components panel in the flash authoring environment. Select window > components in the top menu or press CTRL + F7 to show the components panel.

To use a component, simply drag it onto the Flash stage. The Adrime Components have a visual representation when used in 'Authoring Environment', but not when published to the standard SWF file type. To get the best visual representation, make sure you've got the 'Live Preview' switched on. Enable this by selecting Control > Enable Live Preview in the top-menu.

To edit the settings of an Adrime Component, select it on the stage, and go to the 'Component Inspector' panel. To open this panel, select Window > Component Inspector or use SHIFT+F7. Resize the panel until you see all available options.

### 3 About the components

The AdRime Components package consists of six components. Using the components requires a basic knowledge of Adobe Flash software. Actionscript knowledge is not required. In this chapter, the visual representations of the components are described, followed by small Actionscript examples. For more extensive Actionscript examples, see Chapter 4: Using the ScreenAd Library with Actionscript



Live preview of the AdRime Components

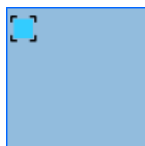
#### 3.1 As Interface component



The AS Interface component gives you access to the Actionscript ScreenAd library. Just drag and drop this to the stage of your flash project to make use of the ScreenAd library in Actionscript. Please see “chapter 4: Using the ScreenAd Library with Actionscript” for setting up your Rich Media campaign with Actionscript.

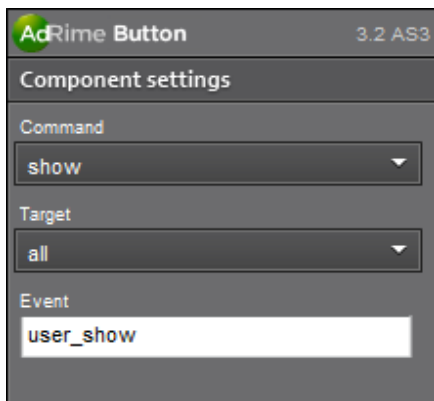
*This component is only needed when you don't use any other component. Once you drag any other AdRime component on the stage, you automatically have access to the entire library.*

#### 3.2 Button component



The button component enables you to quickly and easily create a button. For example, by selecting a predefined command and a target you can create a close button. All commands can be targeted at the Flash object itself or at other creative elements involved in the advertisement.

Drag and drop the button component to the Flash Stage, move and resize the component until it meets the desired dimensions. In the component inspector, you can define the actions (command), the target that should execute the action and the name of the event that will be send to our reporting servers.



Button Component inspector

The following commands and targets are available:

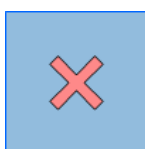
Actions	Description
close	Closes a layer permanently
hide	Hides a layer or a banner
show	Reveals a hidden layer or banner
play	Starts the playhead on the main timeline of the target
stop	Stops the playhead on the main timeline of the target

Targets	Description
self	Targets the object self
banner	Targets the fixed position banner
all	Targets all layers (not banners)
filename_swf	Targets a specific layer or banner. Use the name of the SWF file of this object and replace the dots (.) with an underscores (_).

*Fixed position banners are swf or gif files that are positioned in the default banner position on a websites. E.g. leaderboard, skyscraper or rectangle.*

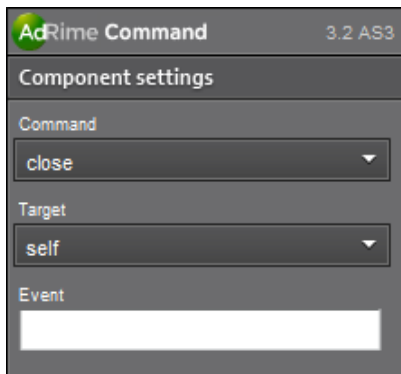
*Layers are swf files that are outside those fixed positions and are 'floating' over the website.*

### 3.3 Command component



The command component executes a command at a certain point in the time. Select the command component and drag it on to the stage at a given point on the timeline. Once the playhead of the Flash player reaches that frame, the command gets executed. The same commands and targets are available as in the button component, the difference with the Command component and the Button component is that the command component will launch commands when triggered, with no click required.

Since the commands are fired directly at a given point in time, the event field is by default empty, which means no event will be fired. If you require an event to be fired, you can enter one in the Event field in the component inspector.



Command component inspector

Use the following Actionscript to send command from Actionscript. This can be done either somewhere on a timeline or as an action behind a button:

*Example 1 Closing all layers (not banners);*

```
screenad.close("all");
screenad.hide("self");
```

*Example 2 Shows a specific layer targeted on filename*

```
screenad.show("layer_500x400_swf");
screenad.play("banner");
```

*Example 3 Stops the playhead on all layers (not the banner);*

```
screenad.stop("all");
```

See chapter 4.7.2 ScreenAd commands for extensive Actionscript about commands.

### 3.4 Click component

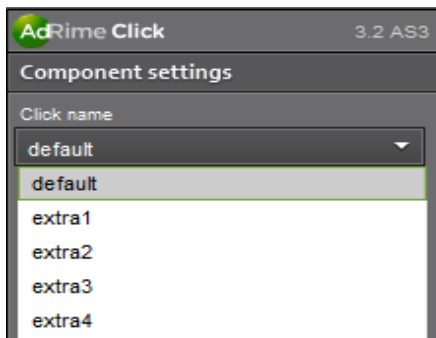


The Click component allows you to define a click(through) area in your movie. After the visitor clicks this area, a new browser window is opened with the targeted website. The component also makes sure the click is registered by the ad management system of the publisher and/or the media agency.

To create a click, just drag the component on stage and resize it until it covers the desired area. You can define multiple exit areas by dragging more components on the stage.

Generally, you have one exit for a flight. It is possible to have multiple exits. The way we work is that the first exit URL will be marked as the “default” URL, and all following clicks will be marked as “extra” clicks. You do not configure the exit URLs in the flash files, instead, you configure them in our previewer environment. This eliminates the need for a standard AS2/AS3 clicktag.

You can define multiple Exit URLs in the Component inspector. Select the click component on the stage and open the component inspector by selecting ‘window’>‘Component inspector’ in the top menu, or by pressing shift + F7.



#### Defining the exit click in the component inspector

Select default for the default exit URL. Choose an extra click to define extra click URLs. The URLs of the actual click targets are defined later on in the Adrime Online Previewer, or when trafficked, in the ad management system.

Use the following code to define a click in Actionsript:

#### Example 4 Defining a default click in Actionsript

```
screenad.click("default"); // or: screenad.click();
```

#### Example 5 Defining an extra click in Actionsript

```
screenad.click("extra1");
```

*The screenad.click command needs to be fired right after the actual click on the click area. If the click command occurs after a certain time after the actual click, a pop-up will occur.*

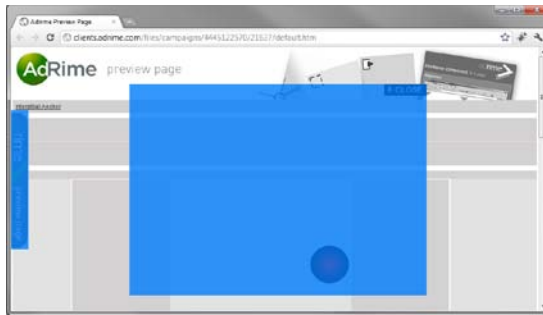
*In case you need to define multiple dynamics clicks loaded from an XML file (for example), please see the Dynamic clicks chapter.*

### 3.5 Clipper component

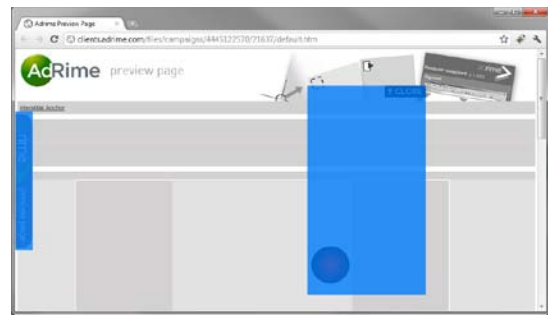


The Clipper component enables you to mask or clip the Flash object on the website.

Sometimes, the content of the flash movie will take up more space, to facilitate the full range of animation. This often means that there's a large part of flash that's not being used, yet still hovers over the website content. This is very bothersome for visitors, since they cannot click on any links underneath the inactive Flash parts. This is where the Clipper component comes in handy, by clipping (or cropping) the ScreenAd to a certain size, you can minimize the amount of space needed during the animation.



Example of a layer without clipping



Example of a layer with clipping

To use the Clipper component, drag it onto the stage and adjust its size so that it marks the area you would like to clip. Everything that does not fall under the clipper will not be shown.

You can easily change the clipping width and height after a certain action or at some point in time by resizing the clipper component.

*Tweening the clipper demands a lot from the computer's processor, we recommend tweening with normal flash masks instead.*

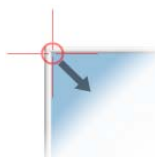
*Do not resize the clipper smaller than 20x20 pixels. Instead, use `ScreenAd.hide()` to hide a layer.*

To use clipping in Actionscript use the following code to create two coordinates to create a clipping rectangle:

#### Example 6 Defining an clipping rectangle

```
screenad.setClip(x1:Number, y1:Number, x2:Number, y2:Number);
```

## 3.6 Positioner component



The positioner component allows you to position your Flash layers on the webpage. Since there's no standard launching position for flash layers on a webpage, all layers must contain the positioner component (or must be positioned using Actionscript). The position of a layer is determined by the alignment setting in the positioner component and the location of the positioner component on the Flash stage.

### 3.6.1 Positioning and aligning a layer

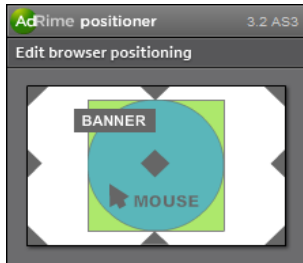
You can set the alignment of a layer by attaching a layer to an anchor point on the web page. Anchor points are the browser corners and edges, the centre of the page, the fixed position ad (or banner) and the mouse cursor. You can select the anchor point by clicking on it in the small browser window in the component inspector panel of the Positioner component. The following alignment options are available:

Horizontal		Vertical	
Left	(edge of the browser)	Top	(edge of the browser)
Right	(edge of the browser)	Bottom	(edge of the browser element)
Center	(of the browser)	Center	(of the browser)
Banner	(the fixed position ad)	Banner	(the fixed position ad)
Mouse		Mouse	

# ID of a HTML element	#ID of a HTML element
------------------------	-----------------------

*When Aligning a layer with the Banner/Banner alignment will position a layer on the TOP-LEFT corner*

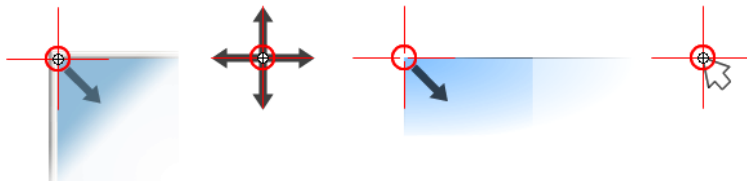
Please keep the different resolutions in mind when aligning a layer.



Anchor points in the component inspector

Above image shows you the available anchor points. The light green area in the small browser window is an estimated preview of the Flash movie layer. The dark grey circle is your Flash content. The dark grey clickable areas represent the different anchor points.

Once you have decided on how to align your layer it is time to position it. Positioning your layer is done by moving the positioner component so that it is on top of the anchor point. The X and Y coordinates of the positioner on the Flash stage are used as offset of the anchorpoint in the browser window.



Four possible Anchor points ( From Left to right: top-left, center, banner and mouse)

*Please note that the positioner component works from its center (that being the middle of the red crosshair), not from the top-left corner of the component. Keep that in mind when positioning layers.*

*Making a screenshot of the website and import it into your Flash project as guide can help you positioning your layer and to determine the offset needed.*

### 3.6.2 Set the size, stage align and the scale mode

In the component inspector it is also possible to set the size of the layer. By adjusting the width and height settings in the component inspector, you can change the size of your layer. This means you are not bound by the width and height of the Flash stage. By default, the width and height are set to 'auto', which means that the width and height of the Flash stage will be used. Aside from this value, you can also enter a fixed amount of pixels or a percentage of the browser size. E.g. if you would like to set a layer to assume the full width of the browser, set the width to 100% and keep the height at 'auto'.

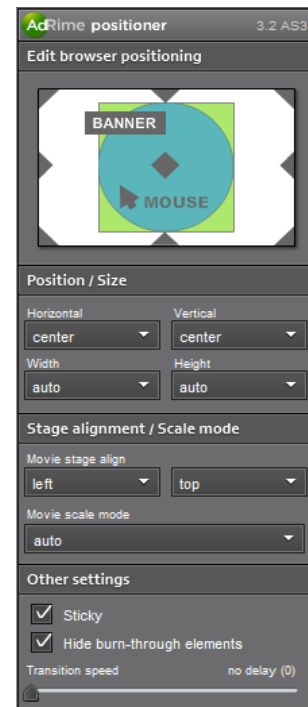
*You can always check the size, scale mode and alignment settings in the small browser window in the component inspector of the Positioner component.*

If the size of your layer is not the same as the size of your Flash movie, it is important to decide how your Flash content should respond to this. You may recognize this from the scale mode and stage alignment of the Flash HTML publishing settings.

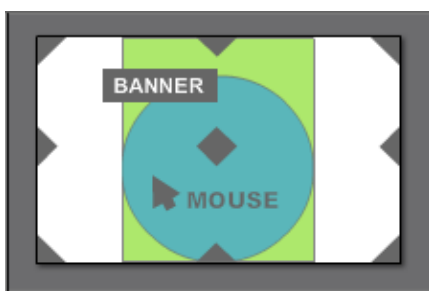
First, you can decide to scale the content of your Flash by setting the Movie scale mode to "show all", "exact fit", or "no borders". You can see the dark green circle in the small browser window scale accordingly to your settings.

The default setting of the scale mode is 'auto'. With this option selected, flash will determine the best scale mode for your project.

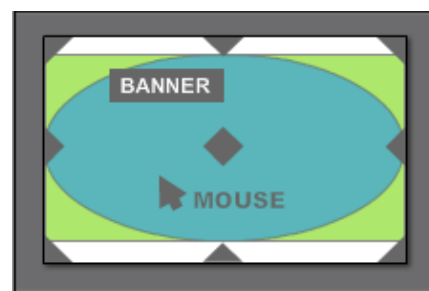
*If the scale mode is set to 'auto', the following rules apply: When the flash animation renders inside the Flash Development Environment, the scale mode of the flash object will be set to 'no scale'. Once running in live mode (inside a browser environment), the scale mode will be set to "Show All". Exceptions are made when the width or height of your layer are set to a percentage value. In that case the scale mode will always be 'no scale'*



When you adjust the Movie stage align settings, the width or height settings and or the scale mode settings, you can see the dark green circle move inside the light green area



width: auto%, height:100%,  
movie stage-align: center, bottom, scale mode: noScale.

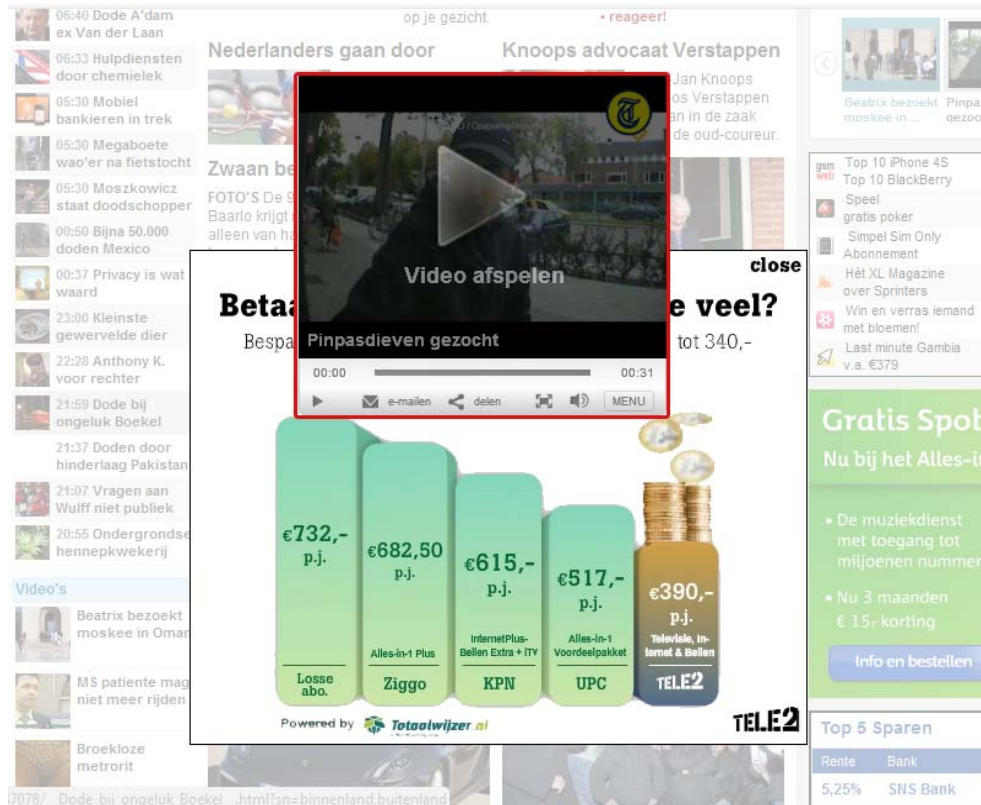


width: 100%, height:auto,  
movie stage-align: center, center, scale mode: exactFit.

### 3.6.3 Other settings: Hide elements And Sticky

Some objects on websites like Video, Flash objects or other plugin object can burn through layers. Turn on the option Hide burn-through elements can solve this problem by hiding the object once the layer is on top of it automatically.

Example 7 Example of a videoplayer burning through a flash layer.



Check the 'Sticky' value if the layer needs to remain in view when the user scrolls down the page. Turn 'Sticky' off to let the layer stay in place when the user scrolls up or down the page.

### 3.6.4 Moving from one position to another

Your layer can move from one position to another by putting another Positioner component in a new key frame. As soon as the play head hits the key frame, the new Positioner settings will be applied.

If you want to smoothen the transition to the new layer position, you can adjust the transition speed, by adjusting the value at the bottom of the window.

*With the transition speed you can create animation when the layer switches from position. It is recommended to only use transitions if you cannot do the animation in your Flash. Transitions can require a lot of processing power.*

## 4 Using the ScreenAd Library with Actionscript

Everything that can be done visually with the Adrime Components can also be done using Actionscript. At least one of the Adrime Components needs to be placed on the stage for the interface to become available. Use the AS Interface if you don't want to use any of the visual components.

### 4.1 The ScreenAd global object

All functionalities of the Actionscript Interface can be accessed through the *ScreenAd* global object.

### 4.2 Position a layer with Actionscript

The positioning of a layer is controlled by three methods:

- ***Alignment***: determines to what anchor point the layer will be aligned. The anchorpoint can be the corners of the browser, the edges of the browser, the center of the browser, the fixed banner position or the mouse position.
- ***Positioning***: Determines with how many pixels offset the layer will be positioned amongst the alignment.
- ***setSize***: changes the size of the flash player.

The Alignment requires two values: horizontal alignment and vertical alignment. The following values are available:

Horizontal	Vertical
left	top
right	bottom
center	center
banner	banner
mouse	mouse

*Example 8 (AS2 & AS3) positioning a layer in the center of the browser with the original Flash movie size (horizontal:center, vertical:center);*

```
screenad.setAlign("center", "center");
screenad.setPosition(-20, -20);
```

Use `setSize` to set the dimensions of a layer. Three kinds of values are possible: the exact amount of pixels, the width and height of the browser in percentages, or the 'auto' option, which uses the size of the flash stage.

*Example 9 (AS2 & AS3) positioning a 100% width layer, left bottom, with a fixed height. (floorad)*

```
screenad.setAlign('left', 'bottom');
screenad.setSize('100%', 'auto');
screenad.setPosition();
```

#### 4.2.1 Positioning elements in a scaling movie

If a Flash element is scaling, for example a layer that always takes the full width of the browser, you can position your content with the `screenad.onResize` handler or the `screenad.RESIZE` event. This way, the position of the content is updated every time the layer changes size.

*Example 10 (AS2) positioning a close button to always display a few pixels from the right side of a movie*

```
screenad.onResize = function() {
    close_mc._x = Stage.width-50;
}
function resizeHandler ( evt:Event ) {
    myclip_mc.x = stage.stageWidth;
    myclip_mc.y = stage.stageHeight;
}
screenad.addEventListener(screenad.RESIZE, resizeHandler);
```

*The `screenad.RESIZE` event will be launched when Flash object changes size. See Chapter 4.8 ScreenAd Events and Properties for more `screenad` Events*

#### 4.2.2 Position transitions

To smoothen the transition from one positioning setup to the next, you can use `setSmooth`. The higher the smoothing number, the slower the transition.

*Example 11 (AS2 & AS3) changing the positioning with a smooth transition:*

```
screenad.setAlign('right', 'bottom');
screenad.setSmooth(10);
```

*It is recommended to only use transitions if you cannot do the animation in your Flash. Transitions can require a lot of processing power.*

#### 4.2.3 Sticky positioning

Sticky positioning means that a layer remains in view when the user scrolls down the page. By default layers are set to be sticky. You can switch it off as shown below:

*Example 12 (AS2 & AS3) switching off sticky positioning:*

```
screenad.setSticky(false);
```

### 4.3 Clipping a layer in Actionscript

With `screenad.setClip` you can clip (or crop) a Flash movie so that unused transparent space doesn't unnecessary cover the web page. *see paragraph 3.5 Clipper component* for more information. You can only clip one rectangular area at a time. The `setClip` function takes the top-left and bottom-right corner coordinates as parameters. 'auto' sets left and top to zero (0), and right and bottom to the width and height of the layer.

*Example 13 (AS2 & AS3) setting the clipper to clip from 0,0 to the width of the layer and 120px height*

```
screenad.setClip(0,0, 'auto', 120);
```

## 4.4 Create an exit click

A click button opens the advertisers web page in a new window. All clicks are recorded by the Adrime statistics system. The URLs can be configured in our previewer environment, so there is no need to set the URL as variables.

*Example 14 (AS2) assigning a click to a MovieClip called "myclip"*

```
myclip.onRelease = function() {  
    screenad.click()  
}
```

When leaving the parameter empty of the click function, the click will automatically refer to the default click.

Multiple clicks to different locations are supported and recorded separately. The first click is called "default", the rest is called "extra1", "extra2",...

*Example 15 (AS3) assigning the third click to a movieclip called "otherclip"*

```
function otherClick( evt:Event ) {  
    screenad.click('extra2');  
}  
  
otherclip.addEventListener( MouseEvent.CLICK, otherClick);
```

### 4.4.1 Dynamic clicks

When loading a lot of exit URLs from a XML file for example, and you want to create dynamic clicks you can use the direct URL instead of the 'default' or 'extra' clicks.

*Example 16 (AS3) using a direct url for a ScreenAd click.*

```
function otherClick( evt:Event ) {  
    screenad.click("http://www.adrime.com");  
}  
  
otherclip.addEventListener( MouseEvent.CLICK, otherClick);
```

*Using a hardcoded url in a flash object is not recommended. Only use this kind of clicktag in combination with, for example, external XML files with a lot of exit URLs that are dynamic.*

### 4.4.2 Setting a wallpaper

You can change the wallpaper (or background) image of the web page and also change its CSS settings (For CSS options go to: [http://www.w3schools.com/css/css\\_background.asp](http://www.w3schools.com/css/css_background.asp))

*Example 17 (AS2 & AS3): setting the wallpaper to wallpaper.jpg*

```
screenad.wallpaper('wallpaper.jpg');
```

*Example 18 (AS2 & AS3) setting the wallpaper to wallpape.jpg and position it fixed in the center of the webpage*

```
screenad.wallpaper('wallpaper.jpg', 'no-repeat fixed center center');
```

To restore the original wallpaper, don't pass any parameters.

```
screenad.wallpaper();
```

## 4.5 Event tracking

Event tracking enables you to track certain events for reporting. This information can be useful to you and the client and is mainly used for measuring interaction rate. You can track expands, user interactions, specific buttons and specific moments in your animation or video.

*Example 19 (AS2 & AS3) tracking when a video has finished playing:*

```
screenad.event('video_complete');
```

The maximum length of an event name is 32 characters. Any events longer than 32 characters will be cut off.

The following characters are allowed in event names: abcdefghijklmnopqrstuvwxyz0123456789 #\*-%:+=?&/,

### 4.5.1 Timer events

It is also possible to start and stop timer events to measure durations of certain interactions. Those timer events will also be shown in the reporting. In order to start a timer use the following code:

*Example 20 (AS2 & AS3) starting a timer event measuring the expand duration*

```
screenad.startTimer("expandDuration");
```

To stop or pause a timer event use:

*Example 21 (AS2 & AS3) stopping or pause a timer event*

```
screenad.stopTimer("expandDuration");
```

## 4.6 Getting information on the environment of the ad

Using the Adrime Components, your Flash movie can get information on its environment or its surroundings on the web page. It knows the location of the mouse pointer, even outside its own space, and it also knows the location and size of the fixed ad position (the banner).

If one of the environment properties changes (for example, the mouse position), the *onEnvironment* event is called, or in Actionscript 3, the *ENVIRONMENT* event.

*Example 22 (AS3) Responding to a change in the environment*

```
function environmentHandler( evt:Event ) {  
    // do something  
}  
screenad.addEventListener(screenad.ENVIRONMENT, environmentHandler);
```

### 4.6.1 Getting the mouse position

#### 4.6.1.1 Mouse position

Using *mouseX* and *mouseY* you can read the mouse position relative to the movie's top-left corner.

*Example 23 (AS2) Responding to the mouse movements, even outside the movie*

```
screenad.onEnvironment = function() {  
    myclip._x = screenad.mouseX;  
    myclip._y = screenad.mouseY;  
}
```

#### 4.6.1.2 Mouse distance

`screenad.mouseDistance()` returns the distance in pixels to the left top corner of the Flash object.

*Example 24 (AS2 & AS3) Getting the distance from the top-left corner of the movie to the mouse*

```
screenad.mouseDistance();
```

#### 4.6.1.3 Mouse angle

`screenad.mouseAngle()`; Returns the angle from the top left corner of the Flash movie to the mouse in degrees. Values arrange from -180 to 180.

*Example 25 (AS2 & AS3) Getting the angle from the top-left corner of the movie to the mouse*

```
screenad.mouseAngle();
```

#### 4.6.2 Getting the banner position

Using `bannerx` and `bannery`, you can read position of the fixed position ad relative to the movie's top-left corner.

*Example 26 (AS2) Placing a movieclip on top of the banner position:*

```
myclip._x = screenad.bannerx;  
myclip._y = screenad.bannery;
```

### 4.7 Setting up communication between flash elements

There are two ways to set up communication between movies: Shared Connection and ScreenAd commands.

The Shared Connection can be used for advanced operations and will work between many Flash movies of the same or a different ad positions (flights) on the same page.

The ScreenAd commands can be used to trigger basic operations on other movies. The command work between all movies of a single flight.

#### 4.7.1 Shared connection

The Shared Connection works with shared methods that can be defined in- and called from any movie. If more than one movie has the same shared method, all these methods will be called at the same time.

*Example 27 (AS2 & AS3) Defining a shared function*

```
screenad.shared.traceSomeText = function( _params:String ) {  
    trace(_params);  
}
```

You can call this shared method from any of the other movies or the movie itself:

*Example 28 (AS2 & AS3) Calling a shared method and passing a parameter*

```
screenad.shared.callMethod('traceSomeText', 'AdRime Manual');
```

Not all movies are loaded at the same time. For example, a layer usually starts loading at a later time than the fixed position banner. To register if all movies are ready for communication, you can use a `onNewElement` event or the Actionscript 3 event `NEW_ELEMENT` that is triggered every time a movie is

added to the Shared Connection and a property *elementCount* that counts the number of connected movies.

*Example 29 (AS2 & AS3) Waiting for three movies to become available*

```
screenad.shared.onNewElement = function() {
    if (screenad.shared.elementCount >= 3) {
        // do something
    }
}
```

#### 4.7.2 ScreenAd commands

ScreenAd commands give you a fast and easy way to set up communication between the movies of a single flight. The commands work with ScreenAd targets, the following command and targets are available:

Actions	Description
close	Closes a layer permanently
hide	Hides a layer or a banner
show	Reveals a hidden layer or banner
play	Starts the playhead on the main timeline of the target
stop	Stops the playhead on the main timeline of the target
Rewind	Sets the playhead of the main timeline of the target to frame 1
setVariable	Sets a variable in the root of the target.

Targets	Description
self	Targets the object self
banner	Targets the fixed position banner
all	Targets all layers (not banners)
filename_swf	Targets a specific layer or banner. Use the name of the SWF file of this object and replace all dots (.) with an underscore (_).

##### 4.7.2.1 ScreenAd commands: close

```
screenad.close( [target:String] );
```

Permanently closes the layer or banner. The target is optional.

##### 4.7.2.2 ScreenAd commands: hide

```
screenad.hide( [target:string] );
```

Hides the target, with the possibility to reveal it later. The target is optional. When hiding the target an event for AS3 will be broadcasted: `screenad.HIDE` and an callback for AS2 is fired: `screenad.onHide`. The property `screenad.isShowing`, will be set to false.

*Property:*

*Example 30 Property `screenad.isShowing` is set to false if the flash object is hiding.*

```
screenad.isShowing;
```

*Callback:*

A callback will be triggered when the hide event is fired:

*Example 31 (AS2) example of a callback when the hide event is triggered.*

```
screenad.onHide = function(){  
    // stop content or video  
}
```

*Example 32 (AS3) example of the event HIDE*

```
screenad.addEventListener(screenad.HIDE, hideHandler);  
  
Function hideHandler(e:Event){  
    // dop something e.g. stop video/audio  
}
```

### 4.7.2.3 ScreenAd command: show

```
screenAd.show( [target:string]);
```

Shows the target, the target is optional. Default target is *self*. After the show command is executed an Actionscript 3 event will be fired: *screenad.SHOW*. In Actionscript 2 the *screenad.onShow* handler will be called. The property *screenad.isShowing* will be set to true.

*Property:*

*Example 33 screenad.isShowing is set to true when the flash object is showing.*

```
screenad.isShowing
```

*Callback:*

A callback will be triggered when the show event is fired:

*Example 34 (AS2) example of the onShow handler*

```
screenad.onShow = function(){  
    // do something: e.g. play();  
}
```

*Example 35 (AS3) example of the screenad.SHOW event*

```
screenad.addEventListener(screenad.SHOW, showHandler);  
  
function showHanlder(e:Event){  
    //do something: e.g. play();  
}
```

### 4.7.2.4 ScreenAd command: play

Starts the playhead on the main timeline of the target.

```
screenad.play([target:string]);
```

### 4.7.2.5 ScreenAd command: stop

Stops the playhead on the main timeline of the target

```
screenad.stop([target:string]);
```

### 4.7.2.6 ScreenAd command: rewind

Sets the playhead on the main timeline of the target to frame 1.

```
screenad.rewind([target:string]);
```

Putting your animations on the 2<sup>nd</sup> frame of the main timeline lets you easily create replay functionality with `screenad.rewind`.

#### 4.7.2.7 ScreenAd command: setVariable

Sets a variable with specified name in the root of the target. The target is optional.

```
screenAd.setVariable( name:String, value:String, [target:String] );
```

Note that the value is always a string type.

### 4.8 ScreenAd Events and Properties

You can add eventListeners that listen to certain ScreenAd events. Those can come in handy to create more interaction in you advertisement. The example below shows how to add a ScreenAd eventListener in Actionscript 2 and Actionscript 3.

*Example 36 (AS3) example for how to add an eventListener*

```
screenad.addEventListener(screenad.ENVIRONMENT, environmentHandler)

function environmentHandler(e:Event){
    // do something on environment
}
```

*Example 37 (AS3) example for how to add an eventListener*

```
screenad.onEnvironment = function(){
    // do something on environment
}
```

The following ScreenAd Flash events are available:

Actionscript 2	Actionscript 3	Property
screenad.onEnvironment	screenad.ENVIRONMENT	
screenad.onResize	screenad.RESIZE	
screenad.onPreloadComplete	screenad.PRELOAD_COMPLETE	screenad.isPreloaded
screenad.onButtonClick	screenad.BUTTON	
screenad.onButtonOut	screenad.BUTTON_OUT	
screenad.onButtonOver	screenad.BUTTON_OVER	
screenad.onClick	screenad.CLICK	
screenad.onClickOut	screenad.CLICK_OUT	
screenad.onClickOver	screenad.CLICK_OVER	
screenad.onVisibility	screenad.VISIBILITY	screenad.isVisible
screenad.onShow	screenad.SHOW	screenad.isShowing
screenad.onHide	screenad.HIDE	screenad.isShowing
screenad.onMouseLeave	screenad.MOUSE_LEAVE	
screenad.shared.onNewElement	screenad.shared.NEW_ELEMENT	

#### 4.8.1 screenad.ENVIRONMENT

The *screenad.ENVIRONMENT* / *screenad.onEnvironment* event is dispatched when the environment of the ScreenAd changes. If the mouse position changes (even outside the flash stage) the environment event will be fired.

See 4.6 *Getting information on the environment of the ad* for more information.

#### 4.8.2 screenad.RESIZE

The *screenad.RESIZE* / *screenad.onResize* event is dispatched when the dimensions of the web browser changes.

See 4.2.1 *Positioning elements in a scaling movie* For more information

#### 4.8.3 screenad.PRELOAD\_COMPLETE

the *screenad.PRELOAD\_COMPLETE* / *screenad.onPreloadComplete* event is fired when the flash is done preloading. When done preloading the property *screenad.isPreloaded* will be set to *TRUE*

With the following code example you can set the amount of percentage that should be loaded before the event is fired:

*Example 38 (AS2 & AS3) setting the preload percentage*

```
screenad.setPreloadPercentage(100);
```

*By default the preload percentage is set to 80%. An event will be fired when preloading reaches 80% of the total amount of bytes loaded.*

#### 4.8.4 screenad.BUTTON

the *screenad.BUTTON* / *screenad.onButtonClick* event is dispatched when the button component receives a click.

#### 4.8.5 screenad.BUTTON\_OUT

the `screenad.BUTTON_OUT / screenad.onButtonOut` event is dispatched when the mouse leaves the button component.

#### 4.8.6 screenad.BUTTON\_OVER

the `screenad.BUTTON_OVER / screenad.onButtonOver` event is dispatched when the Mouse hovers over the button component.

#### 4.8.7 screenad.CLICK

the `screenad.CLICK / screenad.onClick` event is dispatched once an exit click is generated through the Adrime Click component or through `screenad.click`.

#### 4.8.8 screenad.CLICK\_OVER

the `screenad.CLICK_OVER / screenad.onClickOver` event is dispatched when the mouse hovers over the Click Component.

#### 4.8.9 screenad.CLICK\_OUT

the `screenad.CLICK_OUT / screenad.onClickOut` event is dispatched when the mouse leaves the click component.

#### 4.8.10 screenad.VISIBILITY

the `screenad.VISIBILITY / screenad.onVisibility` event will be fired when the visibility of the advertisement changes.

If the advertisement is more than or equal to 50% in the browser window then the property `screenad.isVisible` will be set to `TRUE`.

If less than 50% of the ad is displayed in the browser window or if no interaction with the page has been measured for a certain time, the property `screenad.isVisible` will be set to `FALSE`

*Visibility is not the same a SHOW or HIDE event. A layer can be showing but not be visible at the same time!*

#### 4.8.11 screenad.SHOW

The `screenad.SHOW / screenad.onShow` event will be fired when the banner or layer receives the show command.

The property `screenad.isShowing` will be set to `TRUE`.

#### 4.8.12 screenad.HIDE

the `screenad.HIDE / screenad.onHide` event will be fired when the banner or layer receives the hide command.

The property `screenad.isShowing` will be set to `FALSE`.

#### 4.8.13 screenad.MOUSE\_LEAVE

the *screenad.MOUSE\_LEAVE* / *screenad.onMouseLeave* event is dispatched from javascript once the mouse leaves the Flash stage. This enables you to simulate the Actionscript 3 Event.MOUSE\_LEAVE in Actionscript 2.

#### 4.8.14 screenad.shared.NEW\_ELEMENT

The *screenad.shared.NEW\_ELEMENT* / *screenad.shared.onNewElement* event is dispatched when a new Flash object is registered in the shared connection.

The property *screenad.shared.elementCount* is updated by 1.

## 5 Using Flex, FDT or Flashdevelop

Using a development environment like Flex, FDT or Flashdevelop requires you to import the ScreenAd Library. Upon request, we provide the “asInterface.swc” which you can import in your project. Please contact your local support team for the package.

Use the following example code to import and initialize the ScreenAd library:

```
import screenad;

public function constructor():Void{
    if (stage) this.init();
    else this.addEventListener(Event.ADDED_TO_STAGE, init, false, 0, true);
}

protected function init(e:Event):void{
    screenad.GetInstance();
    screenad.root = this;
}
```

Using Flex also requires you to import the asInterface.swc and add the following code to your MXML:

```
Function Init():Void{
    screenad.InitFlex(Application.application)
}
```



## 6 Actionsript reference

### 6.1 Methods

Method	Usage	Description
<u>Angle</u>	<code>screenad.angle(x1:Number,y1:Number,x2:Number,y3:Number)</code>	Calculates the angle between two points
<u>Click</u>	<code>screenad.click(click:String);</code>	Executes an ad-click. (default, extra1, extra2, extraN, or URL);
<u>close</u>	<code>screenad.close([target:String]);</code>	permanently closes the layer
<u>command</u>	<code>screenad.command(command:string, [target:string]);</code>	Executes a ScreenAd command
<u>Distance</u>	<code>screenad.distance(x1:Number, x2:Number, y1:Number, y2:Number);</code>	Calculates the distance between two points
<u>event</u>	<code>screenad.event(event:String);</code>	Sends an event to the statistics servers
<u>Hide</u>	<code>screenad.hide([target:string]);</code>	Hides the target
<u>MouseDistanceTo</u>	<code>screenad.mouseDistanceTo(x:Number, y:Number);</code>	Calculate the distance between the mouse and a point
<u>Play</u>	<code>screenad.play([target:String]);</code>	Starts the playhead on the main timeline of the target
<u>Postclick</u>	<code>screenad.postclick();</code>	Hides SWF until page receives click
<u>Proceed</u>	<code>screenad.proceed();</code>	Proceed to load next page (only works in combination with postclick)
<u>Resize</u>	<code>screenad.resize(width:Number, height:Number,[target:string]);</code>	Resizes the target



<u>Rewind</u>	<code>screenad.rewind([target:String]);</code>	Rewinds the playhead on the main timeline of the target
<u>setAlign</u>	<code>screenad.setAling(horizontal:String, vertical:String);</code>	Sets the alignment of the layer (left,right,top,bottom,center,banner,mouse,#htmlElementId)
<u>setClip</u>	<code>screenad.setClip(x1:Number, y1:Number, x2:Number,y2:Number);</code>	Sets the clipping area
<u>setHideElements</u>	<code>screenad.setHideElements(hideElement:Boolean);</code>	Turn on to hide elements on the webpage that burns through the advertisement
<u>setPosition</u>	<code>screenad.setPosition(offX:Number, offY:Number);</code>	Sets the position of the layer relative to the alignment
<u>setSize</u>	<code>screenad.setSize(width:string, height:String);</code>	Sets the width and height of the layer;(auto, px, or % values);
<u>setSmooth</u>	<code>screenad.setSmooth(smooth:Number);</code>	Sets smoothing for position changes (0= no smooth, higher number is slower animation);
<u>setSticky</u>	<code>screenad.setSticky(sticky:Boolean);</code>	Switch sticky on or off. If sticky the layer will stay in view when the users scrolls down
<u>setVariable</u>	<code>screenAd.setVariable(varName:string,value:String[,target:string=_ "self"]);</code>	Sets a string variable in the _root of the target
<u>setZIndex</u>	<code>screenad.setZIndex(zIndex:Number);</code>	Sets the ZIndex of the layer
<u>Show</u>	<code>screenad.show([target:string]);</code>	Shows the target
<u>startTimer</u>	<code>screenad.startTimer(timerName:String);</code>	Starts measuring the duration of an event
<u>Stop</u>	<code>screenad.stop([target:string]);</code>	Stops the playhead on the main timeline of the target
<u>stopTimer</u>	<code>screenad.stopTimer(timerNamer:String);</code>	Stop/pause measuring the duration of an event
<u>Wallpaper</u>	<code>screenad.wallpaper([image:String],[css:string]);</code>	Sets a wallpaper on the <body> element. Leave empty to reset
<u>mouseAngleTo</u>	<code>screenad.mouseAngleTo(x:Number, y:Number);</code>	Calculate the angle between the mouse and a point
<u>mouseDistance</u>	<code>screenad.mouseDistance();</code>	Calculate the distance to the mouse



<u>mouseAngle</u>	<code>screenad.mouseAngle();</code>	Calculate the angle to the mouse cursor
<u>setPreloadPercentage</u>	<code>screenad.setPreloadPercentage(preloadPerc:Number);</code>	Sets the preload percentage (defaults to: 80)
<u>Shared.callMethod</u>	<code>screenad.callMethod(methodName:string, [_params:string]);</code>	Calls a shared Method

## 6.2 Events

AS2 Events	AS3 Events	Description
screenad.onButtonClick	screenad.BUTTON	Mousedown on the button component
screenad.onButtonOut	screenad.BUTTON_OUT	Mouseout on the button component
screenad.onButtonOver	screenad.BUTTON_OVER	Mouseover on the button component
screenad.onCLick	screenad.CLICK	Mousedown on the click component
screenad.onClickOut	screenad.CLICK_OUT	Mouseout on the click component
screenad.onClickOver	screenad.CLICK_OVER	Mouseover on the click component
screenad.onEnvironment	screenad.ENVIRONMENT	Is dispatched when the mouse or banner position has changed
screenad.onHide	screenad.HIDE	Is dispatched when the flash object receives a hide event
screenad.onPreloadComplete	screenad.PRELOAD_COMPLETE	Is dispatched when the ad is done preloading
screenad.onResize	screenad.RESIZE	Is dispatched when the ScreenAd (or stage) changed size



screenad.onShow	screenad.SHOW	Is dispatched when the flash object receives a show event
screenad.onVisibility	screenad.VISIBILITY	Is dispatched when the visibility is changed (>=50% in the browserscreen is true, < 50% is false);
screenad.shared.onNewElement	screenad.shared.NEW_ELEMENT	Is dispatched when new Flash file has been registered in the shared connection
screenad.onMouseLeave	screenad.MOUSE_LEAVE	Is dispatched when the mouse leaves the flash stage (dispatched from javascript);

### 6.3 Properties

Properties	Description
<u>screenad.bannerHeight</u>	The height of the banner
<u>screenad.bannerWidth</u>	The width of the banner
<u>screenad.bannerX</u>	X location of the fixed banner position
<u>screenad.bannerY</u>	Y location of the fixed banner position
<u>screenad.Debug</u>	Enable/disable debugging in the output window
<u>screenad.isIde</u>	Returns true if flash is running in the IDE or the flash standalone player
<u>screenad.isPreloaded</u>	Set to true when preloading is complete
<u>screenad.isPreviewer</u>	Returns true when launched within the Adrime previewer environment
<u>screenad.isShowing</u>	Set to true when showing/set to false when hiding
<u>screenad.isVisible</u>	Set To true when >= 50% visible.
<u>screenad.maskClip</u>	Set or unset the clipper as a mask to the stage/root
<u>screenad.mouseX</u>	X Location of the mouse
<u>screenad.mouseY</u>	Y location of the mouse



<u>screenad.scaleMode</u>	Sets the scale mode of the flash movie ( default:auto);
<u>screenad.Shared.elementCount</u>	
<u>screenad.isFlex</u>	True if running in Flex IDE

## 7 Contact

### 7.1 Fixed Support

Any questions about the template, previewer or components, Please contact support:

#### **The Netherlands**

E: [support@adrime.com](mailto:support@adrime.com)

T: +31 (0) 20 52 46 695

#### **Spain:**

E: [support-es@weborama.com](mailto:support-es@weborama.com)

T: +34 91 523 33 30

#### **France:**

E: [adrime@weborama.com](mailto:adrime@weborama.com)

T: +33 (0) 1 53 19 41 82

#### **Italy:**

E: [support-it@weborama.com](mailto:support-it@weborama.com)

T : +39 02 36 60 11 04

#### **United Kingdom**

E: [support-uk@weborama.com](mailto:support-uk@weborama.com)

T: +44 (0)203 538 5773